자동차 네트워크 시스템을 위한 FlexRay 드라이버 개발

The Development of FlexRay Driver for Vehicle Network System

구 용 제*, 김 종 철, 신 충 엽, 박 상 종
(Yong-Je Koo, Jong-Chul Kim, Choong-Yup Shin, and Sang-Jong Park)

Abstract: As the demands for the safety and convenience applications of the vehicles increase, the data load for the In-vehicle Network has increased significantly. As a result, a deterministic and fault-tolerant communication system is required to implement the safety critical applications such as X-by-wire systems. FlexRay communication system is a new standard of network communication system which provides the high speed serial communication, time triggered bus and fault tolerant communication between electronic devices. In addition to time-triggered communication, as providing of the event-triggered communication such as CAN, FlexRay protocol is able to manage the restricted communication resource more effectively. This paper presents the development of FlexRay driver which will be applied to the future ECU's communication system. To develop the FlexRay driver, the FlexRay requirement specification and FlexRay specification is analyzed. The developed FlexRay driver is implemented by using MPC5567 microprocessor of the Freescale semiconductor. To verify the developed FlexRay driver, the waveform of the FlexRay driver was measured and compared with the CAN communication system. As a result, the bus load is reduced about 13% compared with the CAN communication system.

Keywords: FlexRay driver, fault-enhanced algorithm

I. 서론

자동차 안정성 및 편의성에 대한 성능 향상 요구의 증가에 따라 많은 안전 장치가 차량에 탑재되고 있다[4]. 또한 이러한 장치들의 장착 증가에 따른 시스템의 기계적인 복잡도 및 차량 무게 증가에 따른 문제점을 해소하기 위해 기존의 기계적인 연결을 대신하여 전자장치를 이용한 *X*-by-Wire 시스템의 수요가 증가하고 있다. 이와 더불어 증가하는 제어기 사용량의 증가에 따른 통신 네트워크 사용량의 증가와 차량 통신의 내고장성(fault tolerance)의 향상 요구가 증가하고 있다[1,2].

현재 많은 차량용 통신 시스템으로 사용하는 CAN 통신 프로토콜은 통신 네트워크 상의 트래픽이 낮은 경우 우수한 네트워크 성능을 보이지만 트래픽이 높을 경우 모든 메시지의 처리 시간을 보장할 수 없다[1,5].

또한, 전송 지연 편차와 전송 시간의 예측의 어려움으로 *X*-by-wire 시스템에 적용하기 어려운 단점이 있다[2,3].

시분할 방식(TDMA)의 통신 시스템은 한 개 노드의 고장이 다른 노드에 영향을 끼치지 않는 내고장성을 가지며, 트래픽 증가에 따른 데이터 전송의 지연이 거의 발생하지 않는 장점이 있다[2].

FlexRay는 2004년 FlexRay 컨소시엄을 통해 처음으로 규격이 발표되었으며, 그림 1과 같이 현재 사용되는 통신 시스템에 비해 매우 빠른 통신 속도의 구현이 가능하다. FlexRay는 2007년 BMW사의 X5의 서스펜션 제어 장치에 처음 사용되었으며, BMW 7 시리즈 등의 Braking System, AUDI A8의 파워트레인 등에 적용되었다[6-8].

차세대 차량용 통신 네트워크로 각광받고 있는 FlexRay는

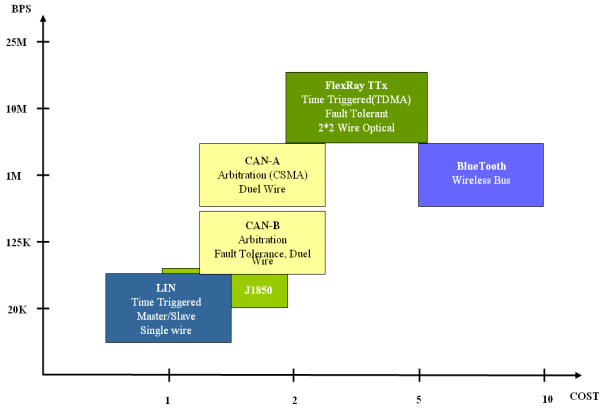

그림 1. 프로토콜 데이터 전송률 비교.
Fig. 1. Comparison of protocol data rates.

시분할 방식의 통신 시스템을 지원하며 최대 데이터 전송률은 10Mbp이며, 두 채널로 동시에 전송이 가능하다[3]. 또한 부가적인 이벤트 트리거 방식의 데이터 전송 방식을 지원함으로써, 시분할 방식의 통신 시스템에서 발생할 수 있는 통신 네트워크 자원의 비효율성에 대한 대처가 가능하다[1,3].

본 논문에서는 FlexRay 통신으로의 전환에 대비하기 위해 FlexRay 통신 시스템 구현에 필요한 FlexRay 드라이버의 구현 및 검증을 수행했다.

II. 통신 프로토콜 분석

1. FlexRay와 CAN 통신의 비교

현재 오토모티브 시장에서 가장 보편적으로 사용하는 CAN과 구현하고자 하는 FlexRay 통신의 특징을 비교하였다. 표 1에서 보는 바와 같이 두 통신 시스템의 특징적인 차이는 전송 속도(baud rate)와 통신 방식이다. CAN 통신은 최대 1Mbps의 데이터 전송 속도로 전송할 수 있는 반면, FlexRay는 각 채널당 10Mbps의 전송 속도를 지원한다. 하지만 통신

* 책임저자(Corresponding Author)
논문접수: 2010. 3. 15., 수정: 2010. 4. 15., 채택확정: 2010. 4. 30.
구용제, 김종철, 신충엽, 박상종: ㈜케피코
(YongJe.Koo@kefico.co.kr/JongChul.Kim@kefico.co.kr/ChoongYup.Shin@kefico.co.kr/SangJong.Park@kefico.co.kr)

표 1. CAN 과 FlexRay 비교.

Table 1. Comparison with CAN and FlexRay.

	CAN	FlexRay
Baud Rate	1 Mbps	10Mbps
Network topology	Bus	Bus & Active Star
Communication Type	Event triggered	Time & Event triggered
Frame ID Length	11/29 bits	11bits
Data Length	8 bytes	254 bytes
Bus Length (meter)	44 (@ 1Mbps)	22 (active star)

네트워크의 기준이 되는 노드 관점으로는 CAN 통신은 한 노드에 한 개 채널만 사용이 가능한 반면, FlexRay 통신은 최대 두 개 채널을 사용 가능하다.

따라서 노드를 기준으로 한 최대 데이터 전송률은 CAN 통신은 최대 1Mbps로 전송이 가능하고, FlexRay는 최대 20Mbps로 전송이 가능하다. CAN 통신은 이벤트 트리거 방식이고, FlexRay 통신은 기본적으로 시간 트리거 방식으로 메시지를 송 · 수신하고, 시간 트리거 방식의 단점인 채널 대역폭 자원 관리의 어려움에 따른 대역폭 낭비를 보완하기 위해 동적 세그먼트에 이벤트 트리거 방식을 추가함으로써 대역폭 손실을 줄일 수 있다[1,2,4].

2. FlexRay 통신 프로토콜 분석

FlexRay는 통신 사이클을 기본 구조로 하며, 아래와 같은 주요 특징을 갖는다.

2.1 통신 사이클(CC: Communication Cycle)의 주요 구성 요소

FlexRay 통신 프로토콜은 통신 사이클(CC)을 기본 단위로 모든 노드가 네트워크 클럭과 동기화 후 통신하는 시분할 방식으로 구현된다. 그림 2와 같이 하나의 통신 사이클은 정적 세그먼트(static segment) 와 동적 세그먼트(dynamic segment), 심볼 윈도우(symbol window), NIT (Network Idle Time)의 4개 영역으로 구분된다[8,9].

2.1.1 정적 세그먼트

정적 세그먼트는 시간 기반의 스케줄링(time-based scheduling)에 의한 통신을 지원하는 영역이다. 정적 세그먼트는 하나의 통신 사이클 내에서 일정한 수의 정적 슬롯으로 구성되며, 슬롯의 수는 모든 노드에서 동일한 값으로 설정되어야 하며, 슬롯의 길이도 모두 동일하게 설정된다[1,2].

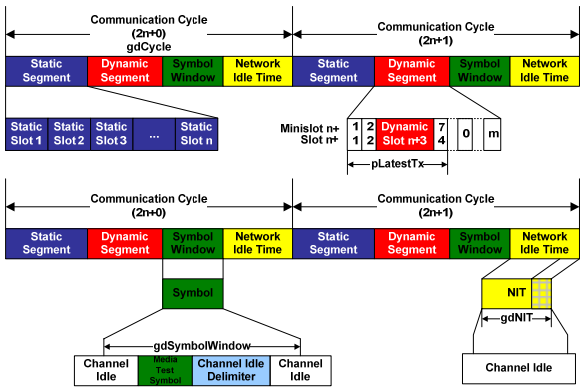

그림 2. FlexRay 통신 사이클의 구조.

Fig. 2. The structure of FlexRay communication cycle.

정적 세그먼트에서 데이터의 송신은 정적 슬롯에 해당되는 시간에 해당하는 슬롯 아이디에 데이터의 송신을 수행하며, 데이터의 수신은 수신 노드에서 수신하고자 하는 시간에 해당하는 정적 슬롯 아이디를 수신 아이디로 설정하여 데이터의 수신을 수행한다[8,9].

2.1.2 동적 세그먼트

동적 세그먼트는 한 통신 사이클에서 이벤트 트리거 방식의 데이터 송수신 지원하는 영역이며, 미니 슬롯(minislot)을 기본 단위로 구성된다. 통신 사이클 내의 미니 슬롯의 수는 각 노드에서 동일한 값으로 설정되어야 하며, 각 미니 슬롯의 길이도 모두 동일하게 설정된다. 그러므로 동적 세그먼트의 길이는 모든 통신 사이클 내에서 동일하다[8]. 만약 불특정 노드에서 데이터 전송 이벤트가 발생하면, 동적 슬롯(dynamic slot)을 사용하여 데이터가 전송되는 방법으로 이벤트 트리거를 지원한다. 동적 슬롯은 전송 요청 시점의 미니 슬롯으로부터 전송하고자 하는 데이터 길이의 미니 슬롯 수로 구성되며, 데이터 프레임 내 페이로드 상위 2바이트로 구성되는 메시지 아이디로 수신 노드를 구분한다[8].

2.1.3 심볼 윈도우

심볼 윈도우 영역에는 충돌 회피 심볼(collision avoidance symbol), Media Access Test Symbol, 웨이크 업 심볼(wakeup symbol)이 있다. 또한 이러한 기능은 선택적으로 사용 유무를 설정할 수 있다[1,2].

2.1.4 Network Idle Time(NIT)

Network Idle Time은 클럭 보정(clock correction)이 수행되는 영역으로, 통신 사이클을 구성하는 하위 단위인 매크로틱(macrotick)의 가변을 통해 (2n+1) cycle마다 NIT 영역에서 클럭 보정이 수행된다[1].

2.2 FlexRay 메시지 프레임의 구조

FlexRay의 메시지 프레임은 그림 3과 같이 헤더, 페이로드, 트레일러의 3부분으로 구성되어 있으며, 소프트웨어는 헤더

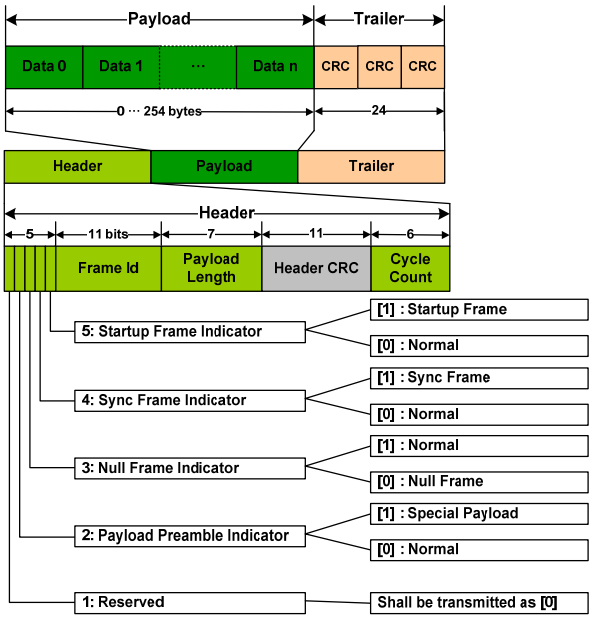

그림 3. FlexRay 프레임 헤더의 구조.

Fig. 3. Header Structure of FlexRay Frame.

및 페이로드에 들어갈 데이터를 생성하여 입력할 수 있어야 한다[1,2].

2.2.1 헤더(header)

FlexRay의 메시지 헤더는 sync, null, startup frame 여부를 나타내는 지시자(indicator) 부분과 데이터 페이로드 길이(data payload length), Header CRC, Cycle Count등으로 구성되어 있으며, 지시자와 페이로드 길이, Header CRC는 사용자에 의해서 결정된다[1].

2.2.2 페이로드와 트레일러(payload & trailer)

설정된 데이터 길이와 동일하게 페이로드는 구성되며, 트레일러에서는 페이로드를 이용하여 연산된 순환 중복 검사(CRC) 값으로 구성된다. 페이로드의 최대 길이는 254바이트까지 가능하며, 헤더의 페이로드 길이 설정을 따른다. 트레일러를 구성하는 순환 중복 검사는 8bits씩 총 3개로 하드웨어에서 연산된다[1].

III. 요구 사항 분석

본 논문은 향후 고객의 FlexRay 드라이버 개발 요청에 대응하기 위한 선행 개발로써 기존 CAN 통신 시스템을 FlexRay 통신 시스템으로 대체하는 것을 목표로 하였다, 이를 위해 예상되는 요구 사항을 분석할 필요가 있으며, FlexRay 요구사항 스펙(FlexRay requirement specification)과 FlexRay 프로토콜 스펙(FlexRay protocol specification) 등을 참조하여, 예상 요구 사항 분석을 하였다[8,10].

소프트웨어 구성에 필요한 요구 사항은 execute normal communication cycle, 통신 모듈 설정(configure communication module), 통신 모듈 종료(shutdown communication module), 클락 동기화(clock synchronization), 웨이크-업 패턴 확인(check for wake up pattern) 등으로 구성할 수 있다[10].

1. 통신 모듈 설정(configure communication module)

해당 요구사항은 Wakeup pattern의 감지나 POC: Halt 상태로 진입 후 또는 어떠한 시간에 POC: Ready 상태로 진입 요청을 받았을 때, 어떠한 정적 혹은 동적 슬롯이 논리적으로 연결되어 전송될지에 대한 시간 스케줄링에 필요한 전역 프로토콜 파라미터(global protocol parameter) 및 기본적인 통신 프로토콜 설정 데이터와 전송 및 수신을 하기 위한 슬롯의 아이디, 메시지의 길이 설정 기능이 필요하다.

1.1 클락 동기화(clock synchronization)

FlexRay 통신의 기본 개념은 시간 트리거 방식이며, 이러한 방식의 통신 시스템을 사용하기 위해서는 클럭 동기화 작업이 반드시 필요하다. 하나의 노드에 연결된 FlexRay 통신 모듈은 통신 네트워크 내부의 요인 뿐만 아니라 실제 온도와 전압 등에 발생할 수 있는 클럭의 오차를 보정하기 위해, sync frame의 수신 여부를 확인할 수 있어야 하며, sync frame의 전송이 필요할 시, 이를 전송할 수 있어야 한다.

1.2 웨이크-업 패턴 확인(check for wake up pattern)

FlexRay 프로토콜의 저전력 모드에서 논리적인 통신 네트워크 IDLE 상태로 진입할 수 있으며, IDLE 상태에서 POC: Ready 상태 천이를 요청을 위한 웨이크-업 패턴 전송에 필요한 웨이크-업 심볼의 생성 및 전송이 가능해야 한다. 또한, 외부 제어기로부터 IDLE 상태의 제어기에 웨이크-업 패턴을 수신 여부를 판별하기 위한 모니터링 기능이 필요하다.

1.3 Execute normal communication Cycle

Execute normal communication Cycle은 하위 요구 사항으로 Receive communication elements, Transmit communication elements, Perform additional Services으로 구성된다.

1.3.1 Receive Communication elements

Receive Communication elements 는 수신된 데이터 스트림의 정보를 이용하여, 호스트 역할을 하는 제어기와 논리적 링크 간의 상태 정보를 주고 받으며, 이 정보를 이용하여 호스트는 수신 데이터 스트림 내의 프레임의 유효성 판별, 버스 상태, 순환 중복 검사(CRC)를 통한 정상 데이터 프레임의 수신 여부 판별, 동기화 또는 스타트-업 명령어 등의 정상 수신 여부 등을 판별할 수 있어야 한다. 또한, 수신 정보의 정상 수신과 에러 발생 등에 대한 정보를 이를 소프트웨어를 통해 사용자에게 알려 줄 수 있어야 한다.

1.3.2 Transmit Communication elements

데이터 전송을 하기 위해 호스트 역할을 수행하는 제어기는 생성한 페이로드 데이터 및 전송을 위해 필요한 통신 elements등을 취합하여 전송 데이터 스트림을 생성해야 한다. 전송에 필요한 통신 elements에는 정적/동적 세그먼트 전송 영역 구성, 전송 프레임 아이디, 슬로 아이디), 헤더 CRC 생성, 스타트-업 프레임, 동기화 프레임 등의 헤더 정보의 설정이 가능해야 한다.

1.3.3 선택적 서비스(optional service)

선택적 서비스에서는 FlexRay 통신을 위해 반드시 구현해야 하는 송/수신 통신 elements와 달리, 기능의 활성/비활성화가 선택적으로 사용 가능해야 하며, 통신 프로토콜 제어 정보를 이용하여 호스트의 에러 정보에 따른 인터럽트 상태 정보, 타이머 설정 및 메시지 아이디 필터링 서비스 등의 구현이 필요하다.

1.3.4 통신 모듈 종료(shutdown communication module)

통신 모듈 종료는 POC: Active_normal 상태에서 호스트로부터 시스템의 종료 요청 수신이나 네트워크 상의 치명적인 에러 발생에 따른 POC: Freeze 상태에 따른 POC: Halt 상태로 진입을 통한 네트워크 종료의 수행이 가능해야 한다.

IV. 설계 및 구현

본 논문에서는 FlexRay 드라이버 구현을 위해 프리스케일(Freescale) 사의 MPC5567을 사용하여 드라이버를 구현하였다. FlexRay 드라이버의 기능 별 구조는 그림 4와 같이 크게 드라이버 환경 설정(configuration), 에러 상태 모니터링(monitoring error status), 프로토콜 명령(protocol command) 등으로 구분할 수 있다[13].

1.FlexRay 드라이버 기능적 분류

1.1 드라이버 환경 설정

통신 환경을 위한 프로토콜 설정(protocol configuration)과 실제 통신 환경에서 송 · 수신하기 위한 데이터에 관한 정보를 설정하는 메시지 설정(message configuration)으로 구성된다. 프로토콜 설정은 FlexRay 모듈의 활성/비활성화, 데이터 전송률, 싱글/듀얼 채널 모드(single/dual channel mode), 채널 별 사용 여부를 설정할 수 있다. 메시지 설정은 송 · 수신 메시지의 최대 길이, 송신 메시지의 종류, 송신 프레임 아이디 등으로 구성이 되며, 수신의 경우 수신 메시지 프레임 아이디와

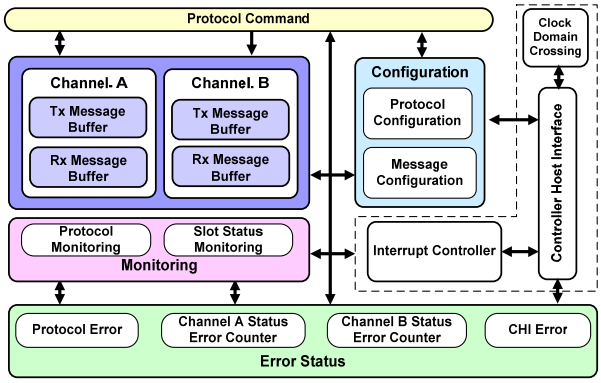

그림 4. FlexRay 드라이버 구조.

Fig. 4. The Structure of FlexRay Driver.

저장 위치 등으로 설정 가능하다.

1.2 모니터링(monitoring)

모니터링 기능은 전체 통신 네트워크의 상태를 확인하는 프로토콜 모니터링(protocol monitoring)과 전송 혹은 수신하기 위한 슬롯의 상태를 확인하기 위한 슬롯 상태 모니터링(slot status monitoring)으로 구성된다.

프로토콜 모니터링 기능은 프로토콜 상태, 스타트-업상태, 웨이크-업 상태, 슬롯 모드, 에러 모드 등의 프로토콜의 전체적인 상태뿐만 아니라 FlexRay 네트워크 상에서 이루어지는 콜드 스타트 신호(cold start) 발생 여부, 네트워크 상의 클럭 보정 오류 발생 여부 등을 확인할 수 있으며, 고객이 요청 시에 외부로 나타내어 보여줄 수 있도록 구성하였다.

1.3 에러 상태(Error status)

에러 상태는 크게 CHI Error, 채널 A/B 상태 에러 카운터, 프로토콜 에러 등으로 구성하였다. CHI Error는 메시지 프레임 손실, 수신 FIFO, Double Message Buffer Lock Error, System Bus Communication Failure Error 등으로 구성하였다. 에러 카운터는 각 채널 별로 슬롯 상태 모니터링을 통해 동적 슬롯, 정적 슬롯, 심볼 윈도우, NIT 등에서 SyntaxError, ContentError, BViolation, TxConflict 등의 발생 여부 및 에러 발생 횟수를 파악할 수 있도록 구성하였다.

1.4 Protocol Command(POCCMD)

POCCMD 는 FlexRay Protocol 상에서 발생하는 프로토콜동작 상태 확인 및 상태 천이가 필요할 시 해당 POC 명령어를 전송할 수 있는 모듈이다.

실제 FlexRay 통신의 시작 및 데이터의 송· 수신 및 종료를 위해서는 그림 5의 POCCMD 절차를 반드시 준수해야 한다.

POC:Default_config 상태에서 FlexRay 드라이버는 통신에 사용되기 위한 채널 A, B의 활성화, 싱글/듀얼 채널 방식의 사용 여부 및 통신에 사용된 클럭을 설정한다. POC:Config 상태에서 FlexRay 드라이버는 사용될 정적 슬롯, 동적 슬롯의 메시지 버퍼 설정 및 참조 메모리 주소, 통신 동기화 정보 및 각종 인터럽트 등의 활성화 여부를 결정한다. POC:Config 상태가 종료된 후 POC:Ready 상태로 천이하게 되며, 이후 Start up 및 Active normal 상태로 진입함으로써 실제 데이터의 송 · 수신이 발생한다. 또한 FlexRay 드라이버는 POC States를 일정한 주기나 메시지 송 · 수신 이전에 확인하여 통신 네트워크 상의 에러나 메시지 전송 관련 설정 변경 요청에 따른 네

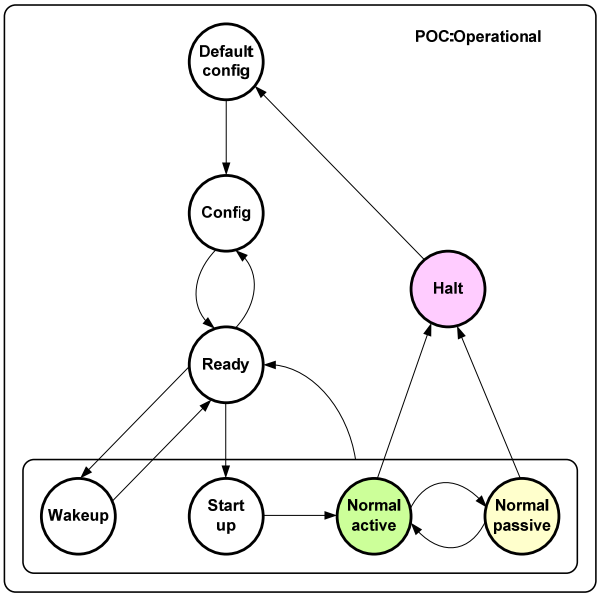

그림 5. FlexRay 드라이버 POCCMD 순서.

Fig. 5. FlexRay driver POCCMD sequence.

트워크 상태 천이가 필요할 시에 해당 상태로 천이하게 된다.

2. FlexRay Driver File Structure & Function

FlexRay 드라이버 소프트웨어는 참고 자료, 표준 문서 등의 분석을 통해 FlexRay 드라이버의 기능 별 분리를 통해, FR_Init.c/h, FR_cfg.c/h, FR_Int.c/h, FR_Util.c/로 구성하였다[9-11,13].

FR_Init.c/h는 FR_cfg.c/h의 설정을 기반으로 마이크프로세서 내부의 프로토콜 관련 레지스터의 초기화 및 설정, 메시지 버퍼의 초기화 등과 같은 실제 통신을 하기 위한 모듈의 초기화 작업을 수행한다.

FR_Int.c/h는 FlexRay 드라이버에서 사용하는 프로토콜 및 송 · 수신 메시지 버퍼 관련 인터럽트 기능을 구현하였으며, 프로토콜 내에서 발생한 각종 에러나 Cold Start, Sync 상태의 발생 여부를 인터럽트를 이용하여 사용자에게 전달할 수 있도록 구성하였다.

FR_util.c/h는 메시지를 송 · 수신하기 위한 기능 및 POCCMD의 전달 기능을 담당한다. 마지막으로 FR_cfg.c/h는 FlexRay 드라이버의 전역 통신 프로토콜 설정, 송 · 수신을 위한 메시지의 길이, 슬롯 아이디, 메시지 버퍼 아이디, 동/정적 세그먼트 사용 여부, 메시지 전송 채널의 선택 등과 같은

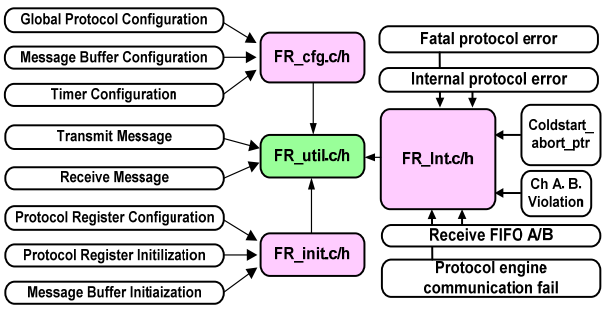

그림 6. FlexRay 드라이버 파일 구조 및 기능.

Fig. 6. FlexRay Driver File Structure and Function.

사용자가 메시지를 전송하기 위해 기본적으로 선택할 수 있는 통신 프로토콜 환경 설정 및 메시지 관련 옵션 등의 설정을 수행할 수 있도록 구현하였다. FlexRay의 통신 프로토콜 관련 환경 설정의 변경 요구가 발생하여도 소프트웨어 내의 추가적인 기능 상의 변경 없이 FR_cfg.c/h의 설정 변경 만으로 각 기능들의 활성/비활성화가 가능하도록 구현하였다.

3. 산출물

FlexRay 드라이버는 FlexRay를 통해 제공될 수 있는 프로토콜 서비스의 분석을 통해 FlexRay 드라이버 개발에 필요한 예상 요구사항의 분석 및 이를 기반으로 프로젝트 별로 요구되는 상이한 서비스들의 유연한 제공을 할 수 있는 구조로 개발하였으며 그 결과 산출물은 다음과 같다.

- 요구사항 분석서
- FlexRay 드라이버의 사양 설명서
- 소프트웨어 구현을 위한 디자인 문서
- FlexRay 드라이버 소프트웨어
- 테스트 사양서, 데이터, 테스트 보고서

V. 검증

구현된 FlexRay 드라이버의 정상 동작의 검증을 수행하였다. 검증을 위한 통신 환경을 구축하기 위해 구현된 FlexRay 드라이버는 FreeScale 사의 MPC5567 Evaluation Board와, FlexRay통신을 지원하는 CANoe 7.0을 이용하여 FlexRay 드라이버의 정상적인 동작을 검증하였다[5].

1. 물리적 계층 검증(physical layer verification)

물리적 계층의 검증을 위해 그림 7과 같이 PC와 MPC5567 간 2개의 통신 노드를 구성하여 테스트 데이터를 전송하였으며, 이를 오실로스코프 파형을 측정하였다.

1.1 Idle, Data 0, Data 1 파형

그림 8은 FlexRay통신의 물리적 계층의 Idle, Data 0, Data 1 등의 3가지 상태를 보여주는 파형이다. 한 채널에는 BP (Bus Plus)와 BM (Bus Minus)가 존재하며, Data 0, Data 1의 경우 Idle 기준으로 BP, BM이 ±0.7V로 벌어지며 데이터를 전송하는 것을 확인할 수 있다.

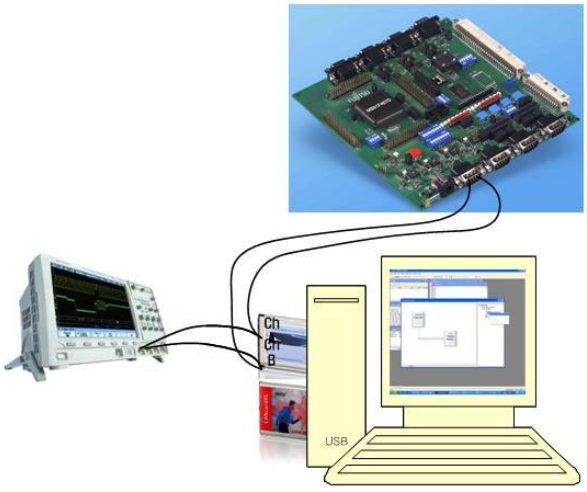

그림. 7. FlexRay Driver 검증 시스템 연결도.
Fig. 7. System Construction for verification of the FlexRay Driver.

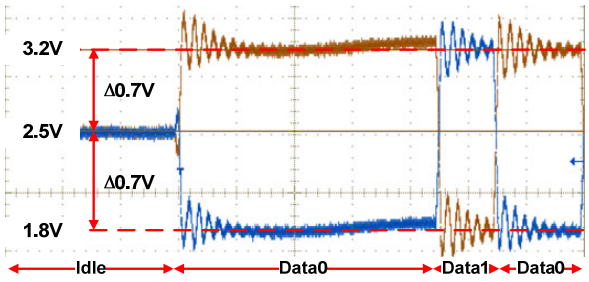

그림 8. Idle, data 0, data1 파형.
Fig. 8. The waveform of the idle, data 0 and data1.

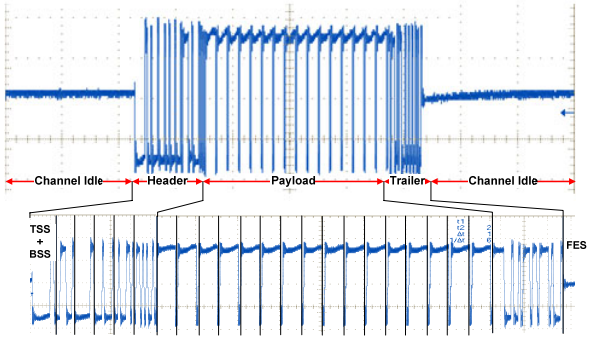

그림 9. 정적 세그먼트 파형.
Fig. 9. The waveform of the static segment.

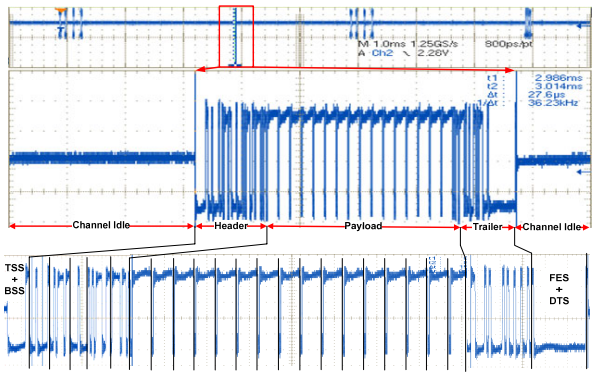

그림 10. 동적 세그먼트 파형.
Fig. 10. The waveform of dynamic segment.

1.2 정적 세그먼트의 파형

앞에서 설명한 바와 같이 헤더, 페이로드, 트레일러가 하나의 정적 슬롯 프레임을 구성한다. 그림 9 파형의 특징을 살펴보면, 신호 사이에 1Byte Sequence (BSS)의 2개 Bit[Data_1→Data_0]을 단위마다 Byte Start 확인할 수 있고, 프레임의 시작 부분에는 TSS (Transmission Start Sequence)와 FSS (Frame Start Sequence)를 확인할 수 있다. 또한 프레임의 마지막 임을 알리는 FES (Frame End Sequence) 2 Bits[Data_0 → Data_1]을 확인할 수 있다.

1.3 동적 세그먼트의 파형

앞에서 살펴본 정적 슬롯 프레임과 동일하게 헤더, 페이로드, 트레일러가 하나의 동적 슬롯 프레임을 구성한다. BSS, TSS, FSS, FES가 모두 정적 슬롯과 동일하지만, 끝 부분의 FES 이후 DTS (Dynamic Trailing Sequence)가 붙어 다음의

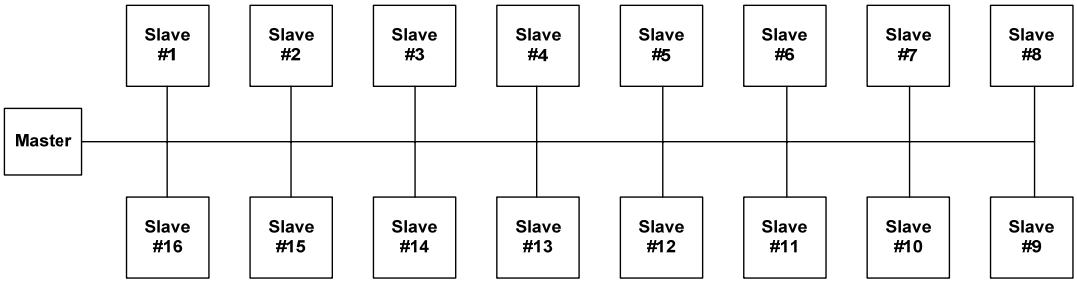

그림 11. Simulation Environment.
Fig. 11. Simulation Environment.

Action Point를 맞추는 구간이 추가로 존재한다. 그림 10 동적 세그먼트의 파형을 통해 정상적인 파형이 출력 됨을 확인 할 수 있다.

2. 드라이버 성능 비교 검증

FlexRay 드라이버의 정상적인 동작과 통신 방식의 효과를 검증해 보기 위해 실제 차량에서 사용되는 다중 제어기간의 CAN 통신 환경과 이를 대체하기 위한 FlexRay 통신 환경을 동일하게 구성하여 통신 부하 측정을 수행해 보았다.

시뮬레이션 대상은 그림 11과 같이 프리스케일 사의 MPC5554 MCU를 사용하는 하나의 상위 제어기와 총 16개의 하위 제어기가 연결되어 동작하는 시스템이며, 제어기 내 처리 알고리즘 대비 통신 데이터 양이 매우 많은 시스템이다. 또한, 데이터의 전송에 있어 데이터 지연에 따른 데이터 처리 지연이 가능한 적어야 하기 때문에 해당 시스템을 선정하였다. 각 하위 제어기는 2개의 프레임 아이디로 데이터를 전송하며, 상위 제어기는 총 32개의 CAN 메시지를 50ms마다 수신한다. 이 때 사용되는 CAN 버스의 부하량은 약 16%를 차지한다.

FlexRay 드라이버의 성능 검증을 위해 상위 제어기를 MPC5567 Evaluation Board로 대체하고, 16개의 하위 제어기들은 CANoe를 통해 데이터를 전송하는 것으로 시뮬레이션 환경을 구성하였다. 표 2와 같이 FlexRay의 최대 전송 속도인 10Mbps 환경에서 실제 데이터 전송 시 통신 버스의 부하양의 비교 결과를 측정하였다. CAN 통신 환경에서 50ms마다 8바이트 길이의 2개의 CAN 아이디를 사용하지만, FlexRay 방식으로는 50ms마다 16바이트를 전송할 수 있도록 구성하였으며, 각 하위 제어기마다 정적 세그먼트의 프레임 아이디를 1개씩 할당하여 전송하도록 구성했다.

시뮬레이션 결과 상위 제어기와 정상적으로 데이터를 수신함을 확인할 수 있었고, 버스 부하는 16%를 차지했던 CAN 통신 방식에 비해 약 13% 정도 감소한 2.67%만을 차지한 것을 확인할 수 있었다.

표 2. FlexRay 프로토콜 파라미터.
Table 2. FlexRay protocol parameter.

Data Rates	10 Mbps
Number of static slot	60
Number of minislot	22
Channel Transmission	Dual Channel
Data length for Static Slot	16 Bytes
Data length for Dynamic slot	8 Bytes

VI. 결론

본 논문에서는 제어기에서 FlexRay 통신을 가능하게 하는 소프트웨어 모듈을 개발하였다. CAN 통신을 사용하는 제어기 간의 통신 시스템의 동작을 그대로 유지하면서 더 넓은 통신 대역폭의 제공이 가능해 졌으며, 또한 통신 네트워크 상의 버스 부하 감소 및 사용 채널 수의 감소 효과에 큰 의의가 있다. 이를 통해 향후 요구되는 차량에서 제어기의 사용 증가에 따른 통신 대역폭 부족 현상에 대처할 수 있을 뿐만 아니라 기존의 CAN통신 방식에서는 구현할 수 없었던 기능인 Dual Channel 모드의 적용이 가능하여 통신 네트워크 상에 문제가 발생시 물리적 재 연결 없이 다른 채널로의 전환이 가능해 네트워크 내부 오류에 대한 통신 안정성을 높일 수 있어 *X*-by-wire 시스템 등에도 적용 가능하다. 또한 통신 서비스 별 기능의 활성화를 용이하게 하여 향후 발생하는 실 차량 네트워크에서 요청되는 서비스의 제공에 쉽게 대응할 수 있도록 구현하였다는 것을 개발 의의로 볼 수 있다.

앞으로 실 차량에서 검증하기 위한 FlexRay 드라이버의 성능 검증에 필요한 다양한 테스트 케이스의 작성을 통해 드라이버의 신뢰성을 높일 수 있도록 할 예정이며, 특정 마이크로 프로세서에 의존적인 부분과 독립적인 부분을 구분하여 다양한 마이크로 프로세서에서 용이하게 구성하여 사용할 수 있도록 개발하는 것이 목표이다.

참고문헌

[1] G. Leen and D. Hefferman, "Digital networks in the automotive vehicle," *IEEE Computer and Control Engineering Journal*, vol. 10, no. 6, pp. 257-266, Dec. 1999.
[2] 윤영환, 장주섭, "*X*-by-Wire System의 개발현황," 한국자동차공학회, 오토저널, pp. 15-20, Feb. 2004.
[3] D. Paret, *Multiplexed Networks for Embedded Systems*, John Wiley & Sons, Ltd, 2007.
[4] 김만호, 이경창, 이석, "*X*-by-Wire 시스템을 위한 통신프로토콜의 성능 평가," 한국 자동차 공학회, pp. 1524-1529, Jun. 2004.
[5] 김만호, 이경창, 이석, "차량용 네트워크 기술연구 동향," 한국 자동차 공학회, pp. 7-14, Sep. 2005.
[6] BMW automobiles, "http://www.bmw.com"
[7] Inautonews, http://www.inautonews.com/bosch-esp-premium-with-flexray-interface, Jul. 2009
[8] FlexRay Consortium, www.flexray.com
[9] FlexRay Consortium, FlexRay Communication System

Protocol Specification V2.1 Rev. A, 2005.
[10] FlexRay Consortium, “FlexRay Communications System Electrical Physical Layer Specification V2.1,” Nov. 2006.
[11] FlexRay Consortium “FlexRay Requirements Specification V2.1,” Dec. 2005.
[12] Robert Bosch Gmbh, “CAN Specification V2.0,” Sep. 1991.
[13] Freescale Semiconductor, “MPC5567 Microcontroller Reference Rev. 1,” Jan. 2007.

구 용 제

2006년 인하대학교 전자전기공학부 졸업. 2008년 인하대학교 대학원 정보공학과 석사 졸업. 2010년~현재 ㈜케피코 재직중. 관심분야는 In-Vehicle Network, Fuel Cell Management System.

신 충 엽

2007년 아주대학교 전자공학부 졸업. 2010년~현재 ㈜케피코 재직중. 관심분야는 Vehicle Network.

김 종 철

2005년 서강대학교 전자공학과 졸업. 2005년~현재 ㈜케피코 재직중. 관심분야는 Diagnostic Communication for vehicle.

박 상 종

2009년 국민대학교 전자공학과 졸업. 2008년 11월~현재 ㈜케피코 재직중. 관심분야는 HEV System.